

Rec'd PCT/EP 10 JAN 2005

PCT/EP 03/07132  
06.08.03

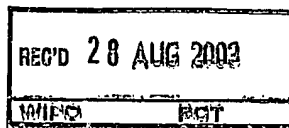
#2



Europäisches  
Patentamt

European  
Patent Office

Office européen  
des brevets



1U/520797

Bescheinigung

Certificate

Attestation

Die angehefteten Unterla-  
gen stimmen mit der  
ursprünglich eingereichten  
Fassung der auf dem näch-  
sten Blatt bezeichneten  
europäischen Patentanmel-  
dung überein.

The attached documents  
are exact copies of the  
European patent application  
described on the following  
page, as originally filed.

Les documents fixés à  
cette attestation sont  
conformes à la version  
initialement déposée de  
la demande de brevet  
européen spécifiée à la  
page suivante.

Patentanmeldung Nr. Patent application No. Demande de brevet n°

02368078.8

PRIORITY DOCUMENT  
SUBMITTED OR TRANSMITTED IN  
COMPLIANCE WITH  
RULE 17.1(a) OR (b)

Der Präsident des Europäischen Patentamts;  
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets  
p.o.

R C van Dijk



Europäisches  
Patentamt

European  
Patent Office

Office européen  
des brevets

PCT/EP 03/07132

06.07.03

Anmeldung Nr:

Application no.: 02368078.8

Demande no:

Anmeldetag:

Date of filing: 11.07.02

Date de dépôt:

Anmelder/Applicant(s)/Demandeur(s):

INTERNATIONAL BUSINESS MACHINES CORPORATION

Armonk, NY 10504

ETATS-UNIS D'AMERIQUE

Bezeichnung der Erfindung/Title of the invention/Titre de l'invention:

(Falls die Bezeichnung der Erfindung nicht angegeben ist, siehe Beschreibung.

If no title is shown please refer to the description.

Si aucun titre n'est indiqué se référer à la description.)

A method and system for extending the file system API

In Anspruch genommene Priorität(en) / Priority(ies) claimed /Priorité(s)  
revendiquée(s)

Staat/Tag/Aktenzeichen/State/Date/File no./Pays/Date/Numéro de dépôt:

Internationale Patentklassifikation/International Patent Classification/  
Classification internationale des brevets:

G06F3/00

Am Anmeldetag benannte Vertragsstaaten/Contracting states designated at date of  
filing/Etats contractants désignées lors du dépôt:

AT BE BG CH CY CZ DE DK EE ES FI FR GB GR IE IT LI LU MC NL PT SE SK TR

## A METHOD AND SYSTEM FOR EXTENDING THE FILE SYSTEM API

### Field of the Invention

The present invention generally relates to file systems; more particularly, this invention concerns a new function of  
5 the file system which facilitates the retrieval of files.

### Background of the Invention

Every operating system includes a file system to manage data files. An API is provided by the operating system to develop applications providing an interface to the user to  
10 manage his own files. A typical application is a file manager to create, move, copy, delete, rename files interactively through a user friendly graphical interface. Other applications such as text or graphical editors also use the API of the operating system to allow the user to manipulate  
15 the data he has processed.

The file system assigns to the users a hierarchical organization of his files. In such a tree data structure, the nodes are folders having branches with other folders, the leaves being the files. Each node and file is identified by a  
20 name which is a character string having a limited size and

that the user is free to assign. The file or folder path is the list of the node names of the system file hierarchy to reach it, the last name being the file or folder name. With the applications using the file system API, the user can first  
5 visualize a hierarchical representation of his files. The graphical view is a tree in which the nodes are identified by the folder and file names. The user selects one folder or file by clicking on the name. The structure depending on one node can be either expanded or collapsed.

10 The applications using the file system API such as the file manager allows first the user to create a file. At the file creation, the file manager proposes the user to give a name. This name is unique for each file. To retrieve an existing file, the user can navigate , through the graphical  
15 representation of the file system hierarchy and select it by clicking on the corresponding node. Retrieving a file knowing its name is immediate. Retrieving a file without knowing its name needs to know its path or at least the name of nodes belonging to its path to find it quickly in the file system  
20 hierarchy.

Usually, when a file is created, the user faces different possibilities to choose the node in the hierarchy where to store it. Most of the time, the file has a relationship with its content, the recipient, the product described in the file,  
25 and multiple other possibilities. At creation time, the recipient information may appear to be the most relevant information and the user chooses to store it in the file hierarchy containing everything related to recipients (customers for example). Unfortunately, today one path must be  
30 selected and only one. At retrieval time, may be months later, the relationship the user remembers might not be the recipient but instead the product described and therefore he will start looking for the file in the hierarchy under products without a chance to find it.

Today, file managers reference a file by its unique combination of path and file name. With the US patent Of Sun Microsystems, Inc., US 6,055,540, a method is disclosed to create a class hierarchy. The patent proposes a document  
5 classification on top of the usual tree data structure built in the file managers, to perform a more efficient retrieval of documents. With the embodiment described, the user can create a class hierarchy by defining a tree structure of category nodes, the leaves and a category definition. The method  
10 provides also an interface to update these category node structure or definition. The solution of the prior art is helpful to be mapped on an application data base such as a software engineering data base used by application developers. It is used, for instance, to organize the access to data  
15 between developers as the categories of data are created in relation with development. This solution of the prior art requires time and process resources to manage the category nodes. It is rather used for coordinating a group of users and to organize the data as they are created and updated by this  
20 group of users.

The standalone user of a computer managing his own file system does not want to create and manipulate a new data structure over the standard tree data structure proposed by the file manager of the operating system. Neither he wants to  
25 store the files according to a list of categories that need to be shared by a group and need to be managed as with the solution of the prior art.

#### Summary of the Invention

It is therefore an object of the present invention to  
30 provide a method to retrieve a file in the standard tree data structure of the file system.

The second object is to achieve the first object by a simple manipulation through a graphical user interface.

These objects are reached by a method, according to claim 1, for managing data using a file name on a computer system having a graphical user interface and a file system storing files with a file hierarchy comprising folders, themselves containing folders or files, said method comprising the steps of:

- entering a command from an application to create a file;
- 10 - displaying the file hierarchy;
- allowing the user to select at least one folder;
- saving data in a first file having the file name in one selected folder;
- in each of the other selected folders, creating a shortcut
- 15 file having the same file name and containing a pointer to the first file;
- creating a hidden file in the folder containing the first file, said hidden file containing the list of pointers to the shortcut files.

20

The method comprises also in the following claims steps for opening, deleting, moving, copying a file created as defined in claim 1. To keep the integrity of the file system, all these file management operations are executed while

25 updating the hidden file which always keeps a track of the occurrences of the file in the file system.

The objects of the invention are also achieved with the use of a program comprising instruction adapted to implement the steps of the method claims when they are executed on a

30 computer.

The object of the invention are also achieved by a computing system comprising means to execute the program as claimed. It is noted that the file management functions,

particularly the multipleStore function, can be integrated in the file system and provided to a computing system user as an extension of the API of the operating system.

5        One advantage of the invention is the use of a virtual link to a real file also called the 'shortcut', 'alias' or LINK according to the operating systems. This function is available with most of the operating system APIs. This function will be generically called shortcut hereafter in the  
10 document.

#### Brief Description of the Drawings

Fig. 1 illustrates a graphical interface of a file manager of the prior art displaying the file system hierarchy  
15 of a user which creates a new file;

Fig. 2 illustrates the user interface file creation operation displaying the 'save multiple' command according to the preferred embodiment;

Fig. 3 illustrates the user interface file creation  
20 operation displaying the file hierarchy while using the 'save multiple' command according to the preferred embodiment;

Fig. 4 is the flow chart of the method of file creation according to the preferred embodiment;

Fig. 5 illustrates the user interface file delete  
25 operation according to the preferred embodiment;

Fig. 6 is the flow chart of the method of file delete operation according to the preferred embodiment;

Fig. 7 is a logical view of the software layers, in a computer, of the software environment of an application implementing the method of the preferred embodiment.

#### Detailed Description of the preferred embodiment

5        Fig. 1 is a screen view of the graphical interface of a file creation operation performed by the graphical editor Freelance Graphics from the IBM Lotus Smart Suite. By selecting the 'Save' command in a menu of actions, a save window (140) is displayed by the application and the user is  
10 requested to save the data he has edited under a file name. The user requires by clicking on a specific icon to display his file system hierarchy. The application opens a second window (100) with the tree representation of the file hierarchy. The user clicks on the folder, here 'Presentation'  
15 (110), of the file system hierarchy in which he wants to save the created file. The user is required to enter the file name (120) and the file type (130).

The user may want to retrieve later his file during the execution of an application using the API of the operating  
20 system. In the same example of Freelance Graphics graphical editor, the user selects the 'Open' command of the menu of actions. An open window is opened by the application and once again the user will click on the specific icon to request his file system hierarchy to be displayed. The user has to  
25 remember the path and the name of the folder 'Presentation' to navigate in the hierarchy. If the user does not remember the name of the folder, he will have to navigate in the file system tree and to open folders. Finally, by clicking on the folder 'Presentation' the user will display the content and  
30 will retrieve his file.

Fig. 2 illustrates the graphical interface to create a file with the method of the preferred embodiment. As with the



prior art, the operation of creating a file is started from any application such as the Freelance Graphics graphical editor. Contrary with the prior art, in the menu of the possible actions, the user can choose between 'Save as' (200) and a 'Save multiple' (210) actions to create a file. When 'Save Multiple' is chosen, the file hierarchy of the file system is displayed as described in Fig. 3. From the graphical representation, the user may select multiple folders according to the relationship he has in mind concerning the data he wants to save in a file. For example, a customer, a product, a technology folder. In the screen view of Fig. 3 three folders have been chosen for the file, the first one being CLIENTA which is the name of a customer for whom the graphical presentation has been created. The 'Save Multiple' operation allows the user to click on a second folder, J2SE because the presentation is related to SUN Java (Java is a trademark of SUN Microsystems) technology. In addition, as the presentation uses the IBM product Websphere, the folder 'Websphere' is also selected. Once the multiple selection is done, the user clicks in the window of the Save button (300) and the application, instead of calling the usual operating system API, invokes a new function (multipleStore (fileName, path1, path2, path3)) which will be described in reference with the flowchart of Fig. 4.

Later on, to retrieve that file from an application such as Freelance Graphics graphical editor, the user will select the 'Open' command in the menu of the possible actions, as with the method of the prior art. The application displays the file hierarchy of the user file system. If the user does not remember the folder in which the file has been saved, he has to navigate in the displayed tree. Now, as the user has created a file under more than one folder, the user has multiplied his chances to remember one of the paths he selected during the creation which had a relationship with the

file content. The user has in mind, for example, to save a presentation prepared for a customer, three relationships:

1. Customer = CLIENTA
2. Technology = J2SE
- 5 3. Product = Websphere

The user selects one of the folders and the files contained in the folder are displayed.

Fig. 4 is the flow chart describing the steps of the method of the preferred embodiment to create a new file using the 'Save multiple' operation. In the first step, the user chooses from the menu of possible actions to save data in a new file (400). A window is displayed in which the user enters the file name. If the user does not choose the 'Save Multiple' (answer No to test 410), the 'Save as' option will be used. This standard option being performed by the application using the usual System file API of the operating system. If the user chooses to perform a 'Save Multiple' (answer yes to test 410), the application reads the file system hierarchy of the user (430) and display the graphical tree structure representation (440). The user selects at least one folder (450) to create the file. Not being part of the method of the preferred embodiment, the user may decide to create folders first if they do not exist. The user is entirely free on his choice of folders. Assuming the user chooses more than one folder, by this choice, the user defines different paths, for instance path1, path2 and path3, for his new file. When the folders are defined, the user clicks on the 'save' button to create the file. The application activates a function (multipleStore (fileName, path1, path2, path3)) which could be integrated as an API of the operating system. The four parameters provided by the application are the inputs entered by the user; they consist in the name of the file and the three paths. The path parameters are coded according to

the operating system, usually as a list of node name with a separator between two items. The multipleStore function reads the first path, path1, and stores the data (460) in a file defined by the fileName and path1, the first path of the parameter list. It is noted that in any other embodiment any of the other folders can be chosen to store the data. Then a shortcut to this first file is created (470) for each of the other paths. In the example two shortcuts to the first file are created, one in the file defined by the fileName and path2, the second in the file defined by the fileName and path3. The shortcut function is part of most of the operating system. It is also called alias or LINK according to the operating system. A shortcut file contains a pointer to one other file. When the user selects a shortcut icon or label on the screen, the system selects the file corresponding to the pointer stored in the shortcut file. One application using the shortcut function is the desktop manager which allows the customer to create shortcut icons of the programs he wants to start execution directly by clicking on the icons from his desktop. The functions supported by the system is the display of a shortcut content (display of the name of the file the shortcut is pointing to), creation, deletion, open and close of a shortcut file. Opening or closing a shortcut file has the effect of opening or closing the real file the shortcut file points to. The shortcut files created in the preferred embodiment are path2/fileName and path3/fileName. In a last step, the function creates a file called shortcutFile (480) at the same location as the file containing the data (path1/fileName in our example). This file is used later during the other file management operation. The function gives to the file system the 'hidden' attributes for the shortcut file; consequently, this file will be hidden when the user displays the tree representation of his user file system. The shortcutFile contains the list of the shortcut files created for this file. In the example the list comprises path2/fileName and path3/fileName.

The retrieval of the file is not part of the invention as being performed by the application using the standard operating system API. When the user of an editor application selects the command 'open' in the menu of possible actions, 5 the file system tree representation is displayed. If the user does not remember the name of the file he navigates in the file system tree until he encounters a folder name reminding him a relationship with the file content. The user to retrieve the file can select anyone of the three folders designated 10 during the Save multiple operation and retrieve the file inside the folder by clicking on the file name inside the selected folder. If the user selects the file physically containing the data (path1/fileName in the illustrative example), the operating system API is used to open this file. 15 If the user selects a shortcut file (path2/fileName or path3/fileName) the shortcut function, which is part of the operating system API, is used to open the real file (path1/fileName).

One variation to this preferred embodiment is, at 20 creation of the file with the 'Multiple Save' command, to let the user the possibility to assign a different file name in each different folder. In the example the multipleStore function could be:

(multipleStore(fileName1,path1,fileName2,path2,fileName3,path3 25 )).

This second embodiment provides a greater flexibility for file naming. The move command which is described later in the document in the case of the preferred embodiment, will be easily adapted by the person skilled in the art man to this 30 second embodiment which allows different file names. As a matter of fact, if the user wants to move a file created by the 'Save Multiple' command to a different folder, the application asks the user if he wants to keep the same file name or not.

The method of the preferred embodiment could also apply to save a file which has already been created using a 'Save Multiple' command according to the preferred embodiment. Assuming that a user has modified such a file, if he activates the 'save' command in the menu of the possible actions, the new version of the file is saved in the real file, even if the user has opened a shortcut of the file. The standard shortcut API is used. If the user activates the 'Save Multiple' command the shortcutFile is read and the tree file system representation is displayed highlighting the folder of the file and the folders of the shortcut files. The user can validate the save as it is by clicking on the save button; he can also de-select or select other folders by clicking on the corresponding tree nodes before clicking on the save button. Correspondingly, new shortcut files are created, other ones are deleted using the shortcut standard API and at the hidden shortcutFile is updated.

The screen view of Fig. 5 illustrates the user interface of the method of the preferred embodiment to delete a file created using the 'Save multiple' command. In the file tree graphical representation (500) displayed by a file manager application, the user selects the file he wants to delete, here 'filesystem2.gif' (510). In the menu of possible actions (520), the user selects the delete function (530).

Fig. 6 shows the flowchart of the method to delete a file. The delete operation is managed with caution to keep the integrity of the system. Assuming with the same illustrative example of Fig. 4 that the user data has been physically saved in a first file (path1/fileName) and that two shortcuts (path2/fileName and path3/fileName) of this first file have been created in two other folders. These shortcuts are also saved in the shortcutFile file.

The user has first selected a folder in the file hierarchy which has been displayed by the application and has selected in the folder the file corresponding to the fileName. Then he has activated the 'delete' command (600) in the menu of the possible actions. The file may have been created using the multipleStore function. A test is performed to check if a hidden file exists in the folder of the first file containing the data. If no hidden file exists (answer No to test 610) the file is deleted (620). If a hidden file exists (answer Yes to test 610), the user is prompted with a warning message 'Delete file from all folders?' (630). The user may decide to remove all the occurrences of the file, the first file, the shortcut files and the hidden ShortcutFile, (answer Yes to test 640). The sequence of operation is, first read the shortCutFile and remove all listed shortcut from the corresponding folders and delete it. At last, the original file and the hidden ShortcutFile are deleted. If the user wants to delete only the occurrence of the file under the folder he has selected, (answer No to test 640) two cases are possible. If the file selected under the fileName is a shortcut file, (answer Yes to test 660), the shortcut file is deleted and this shortcut file address is removed from the hidden shortcutFile (670). If the file selected under the fileName is not a shortcut file (answer No to test 660) but is the first file. This file is moved in a second folder having a shortcut. The corresponding shortcut is removed, the hidden shortcutFile is updated and moved to the second folder (680). In the illustrative example, shortcut pointing to the path2 folder is removed and shortcut to folder path3 is kept.

It is noted that the function of deleting all the occurrences of a file may be optional and that, consequently the warning message 'delete file from all folders' is optionally displayed to the user.

Other operations of a file manager such as 'copy file' or 'move file' are also modified by the method of the preferred embodiment when it applies to file created with the 'Multiple Save' function. The man skilled in the art can adapt the  
5 current commands of the preferred embodiment using the operating system API by adding at the end of the command execution the corresponding update of the shortcutFile.

Fig. 7 illustrates the different software layers in the storage of a computing system operating the method of the  
10 preferred embodiment. In the preferred embodiment, the method is implemented as an application (710) providing the functions described in the document: create file, save, copy, move, delete. The application uses the API of the File system (730), which is a part of the API of the operating system (700). In  
15 the file system API, more particularly, the application uses the shortcut API (720). The application uses the graphical user interface (740) provided with the operating system to interface the user. It is noted that the multipleStore function can be implemented as an extension to the system file  
20 API.

### Claims

1. A method for managing data using a file name on a computer system having a graphical user interface and a file system storing files with a file hierarchy comprising folders, themselves containing folders or files, said method comprising  
5 the steps of:

- entering a command from an application to create a file;
- displaying the file hierarchy;
- allowing the user to select at least one folder;
- 10 - saving data in a first file having the file name in one selected folder;
- in each of the other selected folders, creating a shortcut file having the same file name and containing a pointer to the first file;
- 15 - creating a hidden file in the folder containing the first file, said hidden file containing the list of pointers to the shortcut files.

2. The method of claim 1 further comprising the steps of:

- entering a command from an application to open a file;
- 20 - displaying the file hierarchy and selecting one of the at least one folder;
- selecting the file having the file name;
- if the file to be opened is not a shortcut file, opening the first file;
- 25 - if the file to be opened is a shortcut file, pointing to and opening said first file.

3. The method of claim 1 or 2 further comprising the steps of:

- entering a command from an application to delete a file;



- displaying the file hierarchy and selecting one of the at least one folder;
- selecting the file having the file name;
- if a hidden file does not exist, deleting this file;
- 5 - if a hidden file exists and if the selected file is a shortcut file, deleting the shortcut file and updating the hidden file accordingly;
- if a hidden file exists and if the selected file is not a shortcut file, replacing one of the shortcut file by the
- 10 selected file , updating the hidden file accordingly, moving the hidden file in the folder of the replaced shortcut file and deleting the selected file.

4. The method of claim 3 further comprising the steps of:  
if a hidden file exists:

- 15 - displaying a button to delete selected file having from all folders having conatining a file having the same file name;
- if the button is selected, deleting the selected file and all the other shortcut files or non shortcut file having the same file name and belonging to other folders.

20 5. The method of anyone of claim 1 to 4 further comprising the steps of:

- entering a command from an application to move a file;
- displaying the file hierarchy and selecting one of the at least one folder;
- 25 - selecting the file having the file name;
- selecting a target folder;
- if a hidden file does not exist, moving this file;
- if a hidden file exists and if the selected file is a shortcut file, moving the shortcut file and updating the
- 30 hidden file accordingly;
- if a hidden file exists and if the selected file is not a shortcut file, moving the file to the target folder, updating the hidden file accordingly and moving the hidden file to the target folder.

6. The method of anyone of claim 1 to 5 further comprising the steps of:

- entering a command from an application to copy a file;
- displaying the file hierarchy and selecting one of the at  
5 least one folder;
- selecting the file having the file name;
- selecting a target folder;
- if a hidden file does not exist, copying this file;
- if a hidden file exists, creating in the target folder a  
10 shortcut file of the first file corresponding to the shortcut  
file and updating the hidden file accordingly.

7. A computer program product comprising programming code instructions adapted for executing the steps of the method according to anyone of claims 1 to 6 when said program product  
15 is executed in a computer.

8. A computing system comprising means adapted for executing the computer program of claim 7.

**This Page is Inserted by IFW Indexing and Scanning  
Operations and is not part of the Official Record.**

## **BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☒ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER: \_\_\_\_\_**

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.**